Tutorial: Decision Trees

Building Interpretable Machine Learning Models

ES335 - Machine Learning
IIT Gandhinagar

July 23, 2025

Abstract

Decision trees are among the most intuitive and interpretable machine learning algorithms. This tutorial covers the fundamental concepts, construction algorithms, evaluation metrics, and practical considerations for building effective decision tree models. We'll explore both classification and regression trees, discuss overfitting and pruning strategies, and examine real-world applications through comprehensive examples and exercises.

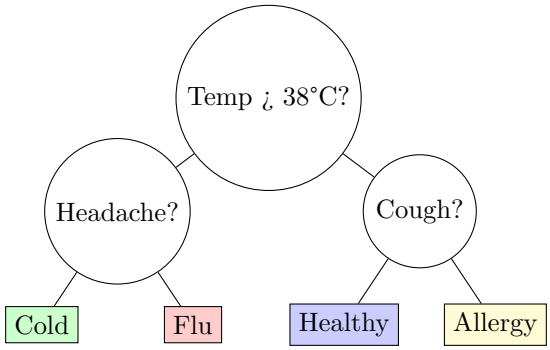
Contents

1 Introduction: Why Decision Trees?

Imagine you're a doctor diagnosing a patient. You might ask:

- Is the patient's temperature above 38°C?
- If yes, does the patient have a cough?
- If yes to both, does the patient have difficulty breathing?

This logical sequence of questions forms a **decision tree** - a flowchart-like structure that makes decisions by asking a series of questions about the input features.



Key Advantages:

- **Interpretability**: Easy to understand and explain
- **No assumptions**: Works with any data distribution
- **Mixed data types**: Handles categorical and numerical features
- **Feature selection**: Automatically identifies important features
- **Non-linear patterns**: Captures complex relationships

2 Mathematical Foundation

2.1 Information Theory Basics

Decision trees work by asking questions that **reduce uncertainty**. We measure uncertainty using information theory concepts.

Entropy measures the "impurity" or uncertainty in a dataset:

$$H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

where p_i is the proportion of samples belonging to class i.

Example #1: Calculating Entropy

Dataset with 8 positive and 2 negative examples:

$$p_+ = \frac{8}{10} = 0.8, \quad p_- = \frac{2}{10} = 0.2$$

$$H(S) = -0.8\log_2(0.8) - 0.2\log_2(0.2) = -0.8(-0.322) - 0.2(-2.322) = 0.722$$

High entropy (close to 1) means high uncertainty, low entropy (close to 0) means low uncertainty.

Information Gain measures how much a feature reduces entropy:

$$\text{Gain}(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

where S_v is the subset of S where feature A has value v.

2.2 Decision Tree Construction Algorithm

The **ID3 algorithm** builds trees by recursively selecting the feature with highest information gain:

1. Calculate entropy of current dataset S
2. For each feature A:
   - Split dataset based on feature values
   - Calculate weighted average entropy of splits
   - Compute information gain
3. Select feature with maximum information gain
4. Create child nodes for each feature value
5. Recursively apply to each child node
6. Stop when: pure node, no more features, or minimum samples reached

Example #2: Building a Tree Step by Step

Tennis Dataset: Play tennis based on weather conditions.

Outlook	Temperature	Humidity	Wind	Play?
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Step 1: Calculate root entropy - 9 "Yes", 5 "No" $\rightarrow H(S) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$
Step 2: Calculate information gain for each feature
For **Outlook**: - Sunny: 2 Yes, 3 No $\rightarrow H = 0.971$ - Overcast: 4 Yes, 0 No $\rightarrow H = 0$ - Rain: 3 Yes, 2 No $\rightarrow H = 0.971$

$$\text{Gain}(\textit{Outlook}) = 0.940 - [\frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971] = 0.247$$

Result: Outlook has highest gain $\rightarrow$ becomes root node!

3 Regression Trees

For continuous target variables, we use **regression trees** that predict numerical values instead of classes.

Key Differences:

- **Split criterion**: Use variance reduction instead of information gain
- **Leaf values**: Mean of target values in leaf
- **Error measure**: Mean Squared Error (MSE) or Mean Absolute Error (MAE)

Variance Reduction:

$$\text{VarReduction}(S, A) = \text{Var}(S) - \sum_v \frac{|S_v|}{|S|}\text{Var}(S_v)$$

where $\text{Var}(S) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$

Example #3: Predicting House Prices

Dataset: House size vs. Price

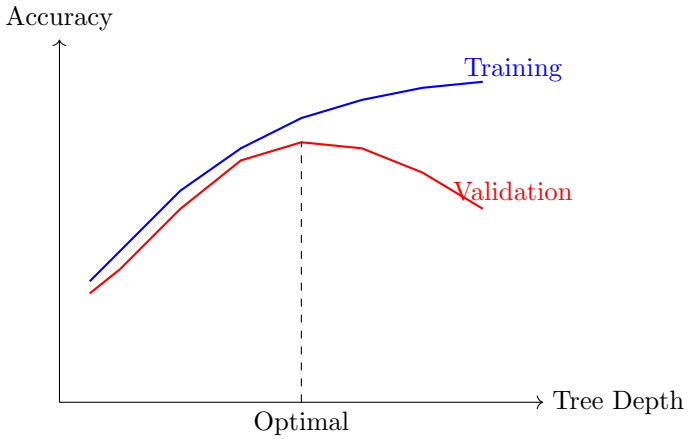
Size (sq ft)	Price ($k)
1000	150
1200	180
1500	220
1800	280
2000	320
2200	350

Possible split: Size ≤ 1500 vs Size > 1500 - Left: Mean = \$183k, Variance = 1156 - Right: Mean = \$317k, Variance = 1344
This split separates smaller, cheaper houses from larger, expensive ones!

4 Overfitting and Pruning

4.1 The Overfitting Problem

Decision trees can grow very deep and memorize training data, leading to poor generalization.



4.2 Pruning Strategies

1. Pre-pruning (Early Stopping):

- Maximum depth limit
- Minimum samples per leaf
- Minimum information gain threshold

2. Post-pruning:

- Build full tree, then remove branches
- Use validation set to evaluate pruning
- Cost-complexity pruning (minimal cost-complexity)

Cost-Complexity Pruning:

$$R_\alpha(T) = R(T) + \alpha|T|$$

where $R(T)$ is the error rate and $|T|$ is the number of leaves. Parameter α controls the trade-off between accuracy and complexity.

5 Advanced Topics

5.1 Handling Categorical Features

For categorical features with k values, there are $2^{k-1} - 1$ possible binary splits. For large k, this becomes computationally expensive.

Strategies:

- **Binary encoding**: Convert categories to binary features
- **Frequency encoding**: Order by target frequency
- **Target encoding**: Use target mean for ordering

5.2 Handling Missing Values

During Training:

- Ignore missing values when calculating information gain
- Use probabilistic splits based on observed data

During Prediction:

- Follow majority path
- Use weighted combination of all paths
- Surrogate splits (backup features)

5.3 Feature Importance

Decision trees naturally provide feature importance scores:

$$\text{Importance}(f) = \sum_{t \in \text{splits using } f} p(t) \times \Delta\text{Impurity}(t)$$

where $p(t)$ is the proportion of samples reaching node t.

6 Implementation Considerations

6.1 Algorithmic Complexity

Training Time: $O(n \times m \times \log n)$ where n = samples, m = features **Prediction Time**: $O(\log n)$ in balanced tree, $O(n)$ worst case **Space Complexity**: $O(n)$ for storing the tree

6.2 Practical Tips

1. **Feature scaling**: Not required (trees are scale-invariant)
2. **Cross-validation**: Essential for selecting optimal hyperparameters
3. **Ensemble methods**: Random Forests and Gradient Boosting improve performance
4. **Interpretability vs. accuracy**: Deeper trees are more accurate but less interpretable

7 Practice Problems

7.1 Basic Problems

Problem #1: Information Gain Calculation

Given this dataset for predicting "Buy Computer":

Age	Income	Student	Buy?
Youth	High	No	No
Youth	High	No	No
Middle	High	No	Yes
Senior	Medium	No	Yes
Senior	Low	Yes	Yes
Senior	Low	Yes	No
Middle	Low	Yes	Yes
Youth	Medium	No	No
Youth	Low	Yes	Yes
Senior	Medium	Yes	Yes

Calculate the information gain for splitting on "Age".
Solution: Root entropy: 6 Yes, 4 No $\rightarrow H = -\frac{6}{10}\log_2(\frac{6}{10}) - \frac{4}{10}\log_2(\frac{4}{10}) = 0.971$
Age splits: - Youth: 1 Yes, 3 No $\rightarrow H = 0.811$ - Middle: 2 Yes, 0 No $\rightarrow H = 0$ - Senior: 3 Yes, 1 No $\rightarrow H = 0.811$
$\text{Gain}(Age) = 0.971 - [\frac{4}{10} \times 0.811 + \frac{2}{10} \times 0 + \frac{4}{10} \times 0.811] = 0.322$

Problem #2: Regression Tree Construction

Build a regression tree for this dataset:

Hours Studied	Exam Score
1	40
2	50
3	60
4	65
5	80
6	85
7	90
8	95

Find the best first split using variance reduction.
Solution: Root variance: Var = 345.31
Test split at Hours = 4.5: - Left (1-4 hours): Mean = 53.75, Var = 91.67 - Right (5-8 hours): Mean = 87.5, Var = 41.67
VarReduction = $345.31 - [\frac{4}{8} \times 91.67 + \frac{4}{8} \times 41.67] = 278.64$
This split significantly reduces variance!

7.2 Intermediate Problems

Problem #3: Pruning Analysis

You have a decision tree with the following validation accuracy at different depths:
Depth 1: 75%, Depth 2: 82%, Depth 3: 88%, Depth 4: 91%, Depth 5: 89%, Depth 6: 85%
a) What is the optimal depth for this tree?
b) What does the accuracy pattern suggest about overfitting?
c) How would you implement early stopping for this case?
Solutions: a) Optimal depth is 4 (highest validation accuracy of 91%) b) Accuracy decreases after depth 4, indicating overfitting to training data c) Set `max_depth=4` or use validation-based early stopping with patience=1

7.3 Advanced Problems

Problem #4: Feature Importance Analysis

Given a decision tree that uses features in this order: [Age, Income, Age, Education, Income], calculate the relative feature importance if each split reduces impurity by [0.3, 0.2, 0.1, 0.15, 0.05] and reaches [100%, 70%, 30%, 20%, 10%] of the training samples.
Solution: Feature importance = $\sum$ (sample_fraction $\times$ impurity_reduction)
- Age: $1.0 \times 0.3 + 0.3 \times 0.1 = 0.33$ - Income: $0.7 \times 0.2 + 0.1 \times 0.05 = 0.145$ - Education: $0.2 \times 0.15 = 0.03$
Normalized: Age: 65.3%, Income: 28.7%, Education: 5.9%

8 Summary and Best Practices

8.1 When to Use Decision Trees

Ideal for:

- Interpretability is crucial

- Mixed data types (numerical + categorical)
- Non-linear relationships
- Feature selection is needed
- Quick prototyping

Avoid when:

- Linear relationships dominate
- High-dimensional sparse data
- Need probability estimates
- Small datasets (prone to overfitting)

8.2 Key Takeaways

1. **Interpretability vs. Accuracy**: Simpler trees are more interpretable but may be less accurate
2. **Overfitting is common**: Always use validation and pruning
3. **Ensemble methods**: Random Forests and Gradient Boosting often perform better
4. **Feature engineering**: Good features are still important for tree performance
5. **Hyperparameter tuning**: max_depth, min_samples_leaf, min_samples_split are crucial

8.3 Implementation Checklist

☐ Use cross-validation for hyperparameter selection

☐ Set appropriate stopping criteria to prevent overfitting

☐ Visualize the tree for interpretability analysis

☐ Calculate and analyze feature importance

☐ Compare with ensemble methods (Random Forest, XGBoost)

☐ Validate on held-out test set

9 Further Reading

- **Classic Paper**: Quinlan, J.R. "Induction of Decision Trees" (1986)
- **Comprehensive Text**: Hastie et al. "Elements of Statistical Learning"
- **Practical Guide**: Kuhn & Johnson "Applied Predictive Modeling"
- **Advanced Topics**: Breiman et al. "Classification and Regression Trees"
- **Modern Implementations**: scikit-learn, xgboost, lightgbm documentation